Linear Regression

ARISE 2020: ECE Machine Learning Lab

Apoorva Nandini Saridena

Department of Electrical Engineering
NYU Tandon School of Engineering
Brooklyn, New York

July 17, 2020

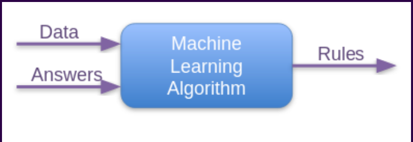
Learning Objectives

- What is the simple linear model for regression?
- What is an error function?
- What is the least squares error function?
- How do we use correlation to tell us about goodness of fit?
- What do we mean by goodness of fit?
- When is it useful to transform the target variable?
- How do we formulate the least squares problem using matrices?
- How does this extend with multi-variable features?
- What is the transformed linear model?
- What do we mean when we talk about "linear"?

Outline

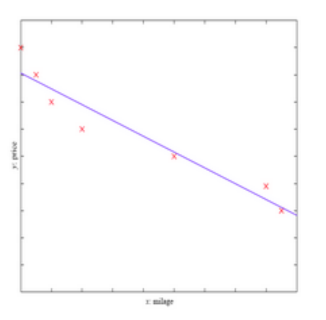
What is Machine Learning

- Train the algorithm from known data to learn the rules
- Make predictions on unknown data using these rules
- Very effective tool where human expertise is not available



Regression

- Target variable is continuous-valued
- Example
  - Predict y = price of a car
  - From x = mileage, size, horsepower
  - Can use multiple predictors
- Assume some form of mapping
  - Ex: Linear mapping: $y = \beta_0 + \beta_1 x$
  - Find parameter β_0, β_1 from data
- Use target-feature pairings as examples to form model



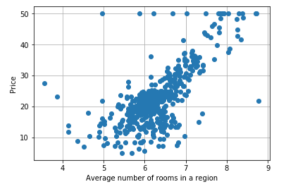
What is Classification?

- Determine what class a target belongs to based on its features
- Example:
  - Predict y = what type of object is in a photo
  - From x = the pixels of the image
- Learn a model/function from features to target
- Use target-feature pairings as examples to form model

Outline

Linear Model

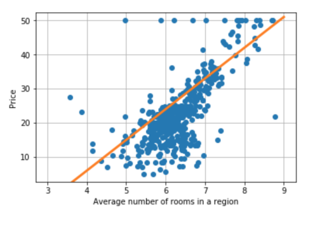
- Data Representation:
  - y = variable you are trying to predict. Also referred to as: Dependent variable, response variable, target variable etc.
  - x = what you are using to predict. Also referred to as: Independent variable, attribute, predictor etc.
  - Set of points, (x_i, y_i), $i = 1, ..., n$. Each data point is called a sample.
  - An efficient way to visualize the data is by plotting y vs x in a scatter plot.



NYU TANDON SCHOOL OF ENGINEERING

Linear Model

- Assume a linear relationship $y = \beta_0 + \beta_1 x$
  - β_0 = intercept
  - β_1 = slope
- $\beta = (\beta_0, \beta_1)$ are the parameters of the model



- Let's do a demo to understand this further.

Outline

Is Your Model a Good Fit?

- How would you determine if your model is a good fit or not?
- Talk with each other to see whose model fits the data the best
  - How will you determine this?
  - Is there a quantitative way?
  - Write python code if so.

Error Functions

- An **error function** quantifies the discrepancy between your model and the data.
  - They are non-negative, and go to zero as the model gets better.
- Common Error Functions:
  - Mean Squared Error: $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$
  - Mean Absolute Error: $MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$
- In later units, we will refer to these as **cost functions** or **loss functions**.
- Compute MSE on your model

General Steps to Solve a Machine Learning Problem

- Load and visualize data

General Steps to Solve a Machine Learning Problem

- Load and visualize data
  - (x_i, y_i), $i = 1, ..., n$

General Steps to Solve a Machine Learning Problem

- Load and visualize data
  - (x_i, y_i), $i = 1, \ldots, n$
- Find an appropriate model to fit the data

General Steps to Solve a Machine Learning Problem

- Load and visualize data
  - (x_i, y_i), $i = 1, ..., n$
- Find an appropriate model to fit the data
  - Eg: Linear model is $\hat{y} = \beta_1 x + \beta_0$

General Steps to Solve a Machine Learning Problem

- Load and visualize data
  - (x_i, y_i), $i = 1, ..., n$
- Find an appropriate model to fit the data
  - Eg: Linear model is $\hat{y} = \beta_1 x + \beta_0$
- Choose an appropriate error function

General Steps to Solve a Machine Learning Problem

- Load and visualize data
  - (x_i, y_i), $i = 1, ..., n$
- Find an appropriate model to fit the data
  - Eg: Linear model is $\hat{y} = \beta_1 x + \beta_0$
- Choose an appropriate error function
  - $MSE = \sum_{i=1}^{N}(y_i - (\beta_0 + \beta_1 x_i))^2$

General Steps to Solve a Machine Learning Problem

- Load and visualize data
  - (x_i, y_i), $i = 1, ..., n$
- Find an appropriate model to fit the data
  - Eg: Linear model is $\hat{y} = \beta_1 x + \beta_0$
- Choose an appropriate error function
  - $MSE = \sum_{i=1}^{N}(y_i - (\beta_0 + \beta_1 x_i))^2$
- Find parameters that minimize the error function

General Steps to Solve a Machine Learning Problem

- Load and visualize data
  - $(x_i, y_i),\ i = 1, ..., n$
- Find an appropriate model to fit the data
  - Eg: Linear model is $\hat{y} = \beta_1 x + \beta_0$
- Choose an appropriate error function
  - $MSE = \sum_{i=1}^{N}(y_i - (\beta_0 + \beta_1 x_i))^2$
- Find parameters that minimize the error function
  - Select β_0, β_1 to minimize the error function

Least Squares Fit

- The **Least Squares Fit** is characterized by the minimization of the MSE error function:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

Least Squares Fit

- The **Least Squares Fit** is characterized by the minimization of the MSE error function:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

  - Find the parameters, $\beta = (\beta_0, \beta_1)$, that give the smallest MSE

Least Squares Fit

- The **Least Squares Fit** is characterized by the minimization of the MSE error function:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

  - Find the parameters, $\beta = (\beta_0, \beta_1)$, that give the smallest MSE
- MSE is a useful metric because there exists an analytic solution to find the optimal parameters β_0 and β_1

Outline

Mean, Variance, and Covariance, Correlation Coefficient

- Given feature-target data
  $(x_i, y_i),\ i = 1, 2, ..., N$

Mean, Variance, and Covariance, Correlation Coefficient

- Given feature-target data $(x_i, y_i),\ i = 1, 2, ..., N$
- Mean:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

Mean, Variance, and Covariance, Correlation Coefficient

- Given feature-target data $(x_i, y_i),\ i = 1, 2, ..., N$
- Mean:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

- Variance:

$$\sigma_x^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2, \quad \sigma_y^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2$$

NYU TANDON SCHOOL OF ENGINEERING

Mean, Variance, and Covariance, Correlation Coefficient

- Given feature-target data $(x_i, y_i),\ i = 1, 2, ..., N$
- Mean:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

- Variance:

$$\sigma_x^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2, \quad \sigma_y^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2$$

- **Covariance**:

$$\sigma_{xy} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})$$

NYU TANDON SCHOOL OF ENGINEERING

Mean, Variance, and Covariance, Correlation Coefficient

- Given feature-target data $(x_i, y_i),\ i = 1, 2, ..., N$
- Mean:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

- Variance:

$$\sigma_x^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2, \quad \sigma_y^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2$$

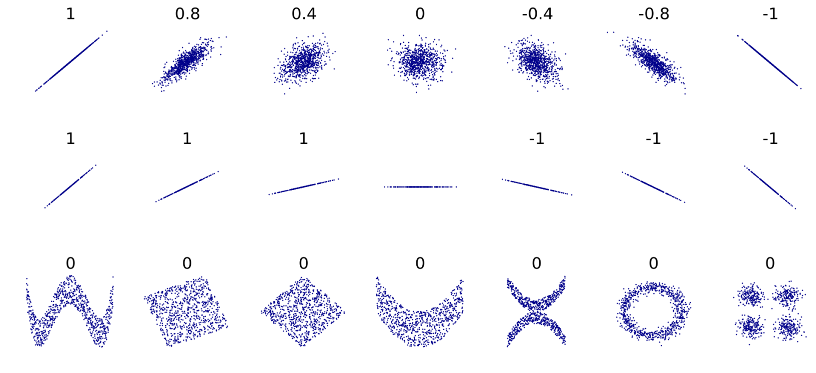
- **Covariance**:

$$\sigma_{xy} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})$$

- **Correlation Coefficient**:

$$\rho = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

Lab: Gaining Intuition



Outline

LS Fit Solution

- Model:

$$\hat{y} = \beta_0 + \beta_1 x$$

- Optimization:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

- Solution:

$$\hat{y} = \bar{y} + \rho\frac{\sigma_y}{\sigma_x}(x - \bar{x})$$

$$\beta_1 = \rho\frac{\sigma_y}{\sigma_x}, \quad \beta_0 = \bar{y} - \beta_1\bar{x}$$

- Prediction:

$$y_{new} = \beta_0 + \beta_1 x_{new}$$

- Compute the LS fit model

Lab: Find/Build and fit your own data

1. Find your a data set
   - Google: "[subject you're interested in] dataset"
   - `https://archive.ics.uci.edu/ml/datasets.php`
   - `https://toolbox.google.com/datasetsearch`

– or –

2. Build your own data set
   - Examples:
     - Use Number of CPUs in a Computer to Predict the Price
     - Number of Instagram Followers to predict likes
   - Only need 10-15 samples

Outline

Extending the Model

- Model: $\hat{y} = \beta_0 \qquad + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_D x_D$

Extending the Model

- Model: $\hat{y} = \beta_0 \qquad + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_D x_D$
- Suppose we have multiple features corresponding to a single target output:

Extending the Model

- Model: $\hat{y} = \beta_0 \qquad + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_D x_D$
- Suppose we have multiple features corresponding to a single target output:
  - y, single target variable

Extending the Model

- Model: $\hat{y} = \beta_0 \qquad + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_D x_D$
- Suppose we have multiple features corresponding to a single target output:
  - y, single target variable
  - multiple features: $x_1, x_2, ..., x_D$

Extending the Model

- Model: $\hat{y} = \beta_0 \qquad + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_D x_D$
- Suppose we have multiple features corresponding to a single target output:
  - y, single target variable
  - multiple features: $x_1, x_2, ..., x_D$
  - Target variable is a linear combination of all feature variables

Extending the Model

- Model: $\hat{y} = \beta_0 \qquad + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_D x_D$
- Suppose we have multiple features corresponding to a single target output:
  - y, single target variable
  - multiple features: $x_1, x_2, ..., x_D$
  - Target variable is a linear combination of all feature variables
  - **Linear Combination**: sum of scaled features

Extending the Model

- Model: $\hat{y} = \beta_0 \quad + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_D x_D$
- Suppose we have multiple features corresponding to a single target output:
  - y, single target variable
  - multiple features: $x_1, x_2, ..., x_D$
  - Target variable is a linear combination of all feature variables
  - **Linear Combination**: sum of scaled features
- Vector Notation:

Extending the Model

- Model: $\hat{y} = \beta_0 \times 1 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_D x_D$
- Suppose we have multiple features corresponding to a single target output:
  - y, single target variable
  - multiple features: $x_1, x_2, ..., x_D$
  - Target variable is a linear combination of all feature variables
  - **Linear Combination**: sum of scaled features
- Vector Notation:

  Let $\mathbf{x} = [1, x_1, x_2, ..., x_D]^T$

Extending the Model

- Model: $\hat{y} = \beta_0 \times 1 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_D x_D$
- Suppose we have multiple features corresponding to a single target output:
  - y, single target variable
  - multiple features: $x_1, x_2, ..., x_D$
  - Target variable is a linear combination of all feature variables
  - **Linear Combination**: sum of scaled features
- Vector Notation:

  Let $\mathbf{x} = [1, x_1, x_2, ..., x_D]^T$
  and $\boldsymbol{\beta} = [\beta_0, \beta_1, ..., \beta_D]^T$

Extending the Model

- Model: $\hat{y} = \beta_0 \times 1 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_D x_D$
- Suppose we have multiple features corresponding to a single target output:
  - y, single target variable
  - multiple features: $x_1, x_2, ..., x_D$
  - Target variable is a linear combination of all feature variables
  - **Linear Combination**: sum of scaled features
- Vector Notation:

  Let $\mathbf{x} = [1, x_1, x_2, ..., x_D]^T$
  and $\boldsymbol{\beta} = [\beta_0, \beta_1, ..., \beta_D]^T$
  So, $\hat{y} = \mathbf{x}^T \boldsymbol{\beta}$

Matrix Formulation

- We want to minimize the squared error over all the samples: $\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{N}(y_i - \mathbf{x}_i^T\beta)^2$

Matrix Formulation

- We want to minimize the squared error over all the samples: $\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{N}(y_i - \mathbf{x}_i^T\beta)^2$
- Design Matrix: Let, $A = \begin{bmatrix} 1 & x_{1_1} & \cdots & x_{1_D} \\ 1 & x_{2_1} & \cdots & x_{2_D} \\ \vdots & & \ddots & \\ 1 & x_{N_1} & \cdots & x_{N_D} \end{bmatrix}$

Matrix Formulation

- We want to minimize the squared error over all the samples: $\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{N}(y_i - \mathbf{x}_i^T\beta)^2$
- Design Matrix: Let, $A = \begin{bmatrix} 1 & x_{1_1} & \cdots & x_{1_D} \\ 1 & x_{2_1} & \cdots & x_{2_D} \\ \vdots & & \ddots & \\ 1 & x_{N_1} & \cdots & x_{N_D} \end{bmatrix}$
- Can express these conditions in matrix form: $\mathbf{y} = A\beta$

Matrix Formulation

- We want to minimize the squared error over all the samples: $\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{N}(y_i - \mathbf{x}_i^T\beta)^2$
- Design Matrix: Let, $A = \begin{bmatrix} 1 & x_{1_1} & \cdots & x_{1_D} \\ 1 & x_{2_1} & \cdots & x_{2_D} \\ \vdots & & \ddots & \\ 1 & x_{N_1} & \cdots & x_{N_D} \end{bmatrix}$
- Can express these conditions in matrix form: $\mathbf{y} = A\beta$
- Solution: Pseudo-Inverse

Matrix Formulation

- We want to minimize the squared error over all the samples: $\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{N}(y_i - \mathbf{x}_i^T\beta)^2$
- Design Matrix: Let, $A = \begin{bmatrix} 1 & x_{1_1} & \cdots & x_{1_D} \\ 1 & x_{2_1} & \cdots & x_{2_D} \\ \vdots & & \ddots & \\ 1 & x_{N_1} & \cdots & x_{N_D} \end{bmatrix}$
- Can express these conditions in matrix form: $\mathbf{y} = A\beta$
- Solution: Pseudo-Inverse
  - There isn't a true solution to this equation.

Matrix Formulation

- We want to minimize the squared error over all the samples: $\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{N}(y_i - \mathbf{x}_i^T\beta)^2$
- Design Matrix: Let, $A = \begin{bmatrix} 1 & x_{1_1} & \cdots & x_{1_D} \\ 1 & x_{2_1} & \cdots & x_{2_D} \\ \vdots & & \ddots & \\ 1 & x_{N_1} & \cdots & x_{N_D} \end{bmatrix}$
- Can express these conditions in matrix form: $\mathbf{y} = A\beta$
- Solution: Pseudo-Inverse
  - There isn't a true solution to this equation.
- We say $\beta^\star$ solves $\mathbf{y} = A\beta$ in the least squares sense, where

$$\beta^\star = A^\dagger \mathbf{y}$$

Boston Housing Demo

- Using multiple features to predicting house prices
  - Crime rate per capita, number of rooms, student-teacher ratio at local schools, ...
- Lets use a linear model that takes into account all the collected data

M6: *Demo*: Multivariable Regression on Boston Housing Data

```
import pandas as pd
import numpy as np

df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data',
                 header=None,delim_whitespace=True,names=names,na_values='?')

df.head(6)  # print the first six samples
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222.0	18.7	394.12	5.21	28.7

NYU TANDON SCHOOL OF ENGINEERING

Outline

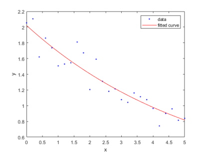
Robot Arm Calibration

- Let's train a model based on the given data.
- In this lab we're going to:
  - Predict the *current* drawn
  - Predictors, X: Robot arm's joint angles, velocity, acceleration, strain gauge readings (load measurement).

Outline

Transforming the Output Data

- Not all data can be modeled using linear relation:
  - $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_D x_D$
- Modeling nonlinear data with linear function does not result in a good fit
- We can transform output with a nonlinear function
- What transformation we use depends on the nature of the data
- For example: We might use an exponential model for:
  - Radioactive decay
  - Population growth



- Demo: mpg of cars data

Outline

Polynomial Regression

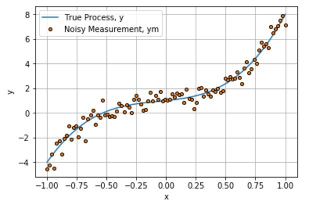
- What if our data is clearly *not* linear, but instead follows a polynomial curve?
  - Ex: Projectile motion, gravity, Coulomb's law, ...
- Can we perform regression to get a polynomial model?
  - Could it be linear regression? (What do we mean by linear?)
  - What would our features be?
- Polynomial Model: $\hat{y} = \sum_{i=0}^{N} \beta_i x^i$
  - Is this linear?

Polynomial Regression

- What if our data is clearly *not* linear, but instead follows a polynomial curve?
  - Ex: Projectile motion, gravity, Coulomb's law, ...
- Can we perform regression to get a polynomial model?
  - Could it be linear regression? (What do we mean by linear?)
  - What would our features be?
- Polynomial Model: $\hat{y} = \sum_{i=0}^{N} \beta_i x^i$
  - Is this linear?

Polynomial Regression

- What if our data is clearly *not* linear, but instead follows a polynomial curve?
  - Ex: Projectile motion, gravity, Coulomb's law, ...
- Can we perform regression to get a polynomial model?
  - Could it be linear regression? (What do we mean by linear?)
  - What would our features be?
- Polynomial Model: $\hat{y} = \sum_{i=0}^{N} \beta_i x^i$
  - Is this linear?



Outline

Transformed Linear Model

- We can extend polynomial fitting to a more general model
- In polynomial fitting, we used the equation:
  - $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_D x_D$
- In the more general model, the output is a linear combination of transformed input
  - $y = \beta_0 + \beta_1 \phi_1(x) + \beta_2 \phi_2(x) + ... + \beta_D \phi_D(x)$
  - where $\phi_1(x)$, $\phi_2(x)$ ..., $\phi_D(x)$ are called **basis** functions
  - Polynomial fitting is a special case where the basis functions are power functions
- Besides polynomials, we can also use other function as our basis function
  - Gaussian: $\phi(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
  - Exponential: $\phi(x) = e^{-\alpha x}$

Transformed Linear Model

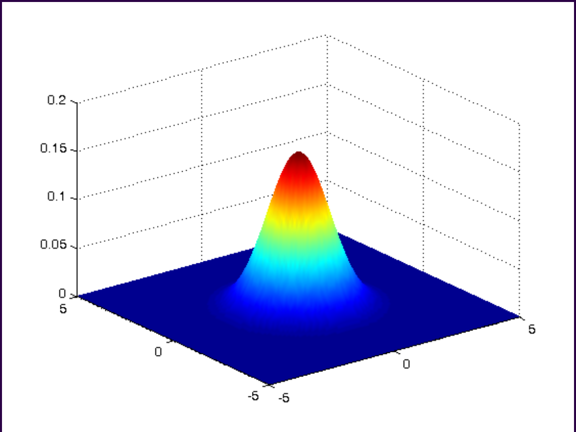
- How do we fit a transformed linear model?
  - $y = \beta_0 + \beta_1\phi_1(\mathbf{x}) + \beta_2\phi_2(\mathbf{x}) + \ldots + \beta_D\phi_D(\mathbf{x})$
- The procedure is similar to polynomial fitting
- First transform the features of each example using the transformation
- Form the Design Matrix: $\Phi = \begin{bmatrix} \phi_0(\mathbf{x}) & \dots & \phi_D(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}) & \dots & \phi_D(\mathbf{x}) \end{bmatrix}$
- Solve for the Least-squares solution:
- $\boldsymbol{\beta} = \Phi^{\dagger} y$
- Model: $\hat{y} = \phi(\mathbf{x})^T \boldsymbol{\beta}$

Transformed Linear Model

- When the input data has multiple features, the $\mathbf{x}$ in the previous equation, $y = \beta_0 + \beta_1\phi_1(\mathbf{x}) + \beta_2\phi_2(\mathbf{x}) + ... + \beta_D\phi_D(\mathbf{x})$, can also be regarded as a vector
- The transformations are then multivariate functions that uses multiple features in generating each new feature
- One example would be the multivariate Gaussian function:
- $\phi(x_1, x_2) = e^{-\frac{(x_1-\mu_1)^2+(x_2-\mu_2)^2}{2\sigma^2}}$ Similar to the 1D Gaussian function which has a bell shaped curve centered at the mean, the 2D Gaussian function is a 2D bell shaped curve centered at (μ_1, μ_2)

Transformed Linear Model

- Shape of a 2D Gaussian function



Outline

Lab: Fitting a curve with Transformed Features

- Open Lab Notebook
- Do the lab in Module 11
- From the plot, think about what function can you use to transform the feature and perform a regression

Learning Objectives

- What is the simple linear model for regression?
- What is an error function?
- What is the least squares error function?
- How do we use correlation to tell us about goodness of fit?
- What do we mean by goodness of fit?
- When is it useful to transform the target variable?
- How do we formulate the least squares problem using matrices?
- How does this extend with multi-variable features?
- What is the transformed linear model?
- What do we mean when we talk about "linear"?

Thank You!

- Next Class: Generalization Error
- How do our models hold up against prediction new data?